End-user programming

‘End-user programming’ is a term that has been introduced to me by the editor of the IEEE magazine *SOFTWARE* in his column, “The dangers of end-user programming” (July / August 2004). Warren Harrison defines end-user programmers as “mechanical engineers, doctors, physicists, teachers, and accountants. They are marketing assistants, receptionists or commodity brokers. They are, in short, individuals who taught themselves to program.” The idea of having ‘end-users’ — people with no background in software testing or development — directly manipulate software that is critical for a company frightens the software professional. Harrison’s conclusion is that end-user programming is a threat for mission-critical systems from the perspective of both correctness and security, and he asks himself:

> *“Can it be true that software manipulating my credit history could have been written by an accountant with no concept of software testing or development processes?”*

Observation 1
Given my observations at several companies, I think that the answer to this question is “yes.” Typical end-user programs are spreadsheets and small database programs. They start small but can evolve into strategic programs for a company.

What is the motivation for end-users to get involved in software development? After all, end-users are educated for a different subject matter; they did not choose an education as programmer, so why would they start building a system, investing so much (free) time in learning how to code in Microsoft Excel, Access, or Visual Basic, and overcome all the barriers involved with end-user programming?

These programs are born out of a need: a businessperson has some particular knowledge that he wants to re-use, automate, or share with colleagues. IT departments seem not to be able to accommodate such needs for reasons that are unclear. When IT does build an application to accommodate the needs of the business, all rules (knowledge) are hard-coded in the programs, business people are not able to inspect the rules, and changes to the business rules take a very long time.

Observation 2
There is a legitimate source of frustration by business people about the software development process.

The business rules approach offers a range of methods and techniques to improve the collaboration between IT departments and business departments. A business rules management environment targeted at end-users is a better idea than end-user programming because this environment should limit the options of an end-user to only change rules, while security aspects, for example, are handled by the business rules management environment. But there is much more needed to succeed.

Observation 3
It is very hard to have end-users create valid, consistent, well-defined, thoroughly-tested business rules. Many business rules end-user maintenance projects failed because of lack of support for these aspects. This problem is not only a problem of ‘end-users’ but of humans in general. Humans have a hard time comprehending more than 20 rules that have numerous complex interdependencies, resulting in rules with undesired effects or that are difficult to maintain and understand.

Given the facts that regular software development environments do not have specialized support for verification and validation of business rules and that communication between business and IT often involves misunderstanding, I observe:

With proper support for verification and validation of business rules in a business rules maintenance environment, the correctness of software with respect to business rules will improve rather than be threatened. The time we gain with serious end-user programming, supported with a good business rules maintenance environment, can be spent by software professionals on the security and stability of the systems supporting the rules of the business.

The business rules approach offers a viable approach to support end-user programming.